MODIS ANNUAL REPORT
- DECEMBER 1992 -

UNIVERSITY OF MIAMI
RSMAS/MPO

DR. ROBERT H. EVANS

NAS5-31362

==========

Due to the interlocking nature of a number of projects, this and subsequent reports will contain coding to reflect the funding source. Modis funded activities are designated with an **M**, SeaWIFS with an **S**, Pathfinder with a **P**, and Headquarters with an **H**. There are several major sections within this report; Database, client/server, matchup database, and DSP support.

A. NEAR TERM OBJECTIVES

A.1 Modis Objectives (M)

A.1.1. Continue to develop and expand the processing environment

a. increase computational efficiency through concurrent operations

b. determine and apply more efficient methods of data availability for processes

A.1.2. Begin extensive testing using global CZCS and AVHRR GAC data with database processing to test the following:

a. algorithm capability
b. machine and operating system stability
c. functionality required for the processing and analysis environment

A.2 SeaWIFS Objectives (S)

A.2.1. Continue testing of processing methodology.

A.2.2. Continue to develop relationship between database and *in-situ* environment.

A.3 Pathfinder Objectives (P)

A.3.1. Expand matchup database as applicable.

A.3.2. Continue testing of methodology.

A.4 DSP Objectives (H)

A.4.1. Continue testing of processing methodology.

A.4.2. Continue to expand the number of sites supported.

A.4.3. Expand the supported hardware/software platforms

B. OVERVIEW OF CURRENT PROGRESS

B.1 Automatic Processing Database (S) (PATHFINDER)

B.1.1 Activity Summary Through November 1992

Extensive changes have occurred in the database-controlled automatic processing system. The major structure of the system remains relatively unchanged, but existing components have been revised and extended, and many new elements have been added. Major elements of the processing system are:

1. MCP - Master Control Program - Actually performs the processing by running the DSP Image Processing system, and calling a set of DSP procedure files.

2. Workstation DSP Image Processing System - Extended to run in UNIX environments.

3. DSP Command Procedure files - Set up the DSP environment variables and call the proper DSP programs.

4. Client/Server information transfer system - Links UNIX and VAX/VMS operating systems.

5. VMS AUTOPROC Database Interface - A set of programs and subroutine that enter and retrieve data into/from the RDB/VMS database.

6. AUTOPROC Database - RDB/VMS Database that holds the information needed to control processing.

7. Miscellaneous control and monitoring programs and command procedures.

Many of these elements (such as the Client/Server and many of the control and monitoring programs/command files) are new; the remainder have all undergone extensive revision in the past year. The major changes, and the reasons they were made, are:

1. MCP

The previous MCP system was a batch job that ran only under the VMS operating system. This had to be changed to allow processing in UNIX environments. Previously, MCP controlled the spawning of other batch jobs in this manner: When it determined that resources permitted the creation of a new job, MCP would contact the database (using the VMS) interface) and request the next file to process. It would spawn a batch job called DBBAT that performed the actual processing. DBBAT then contacted the database independently, and retrieved the first step the processing. When that step was complete, DBBAT again contacted the database, and provided status information (failure or success). If successful, DBBAT retrieved the next step. When all steps were complete, DBBAT reported this back to the database and exited.

In the new system, MCP does not perform job control. One MCP is started for each job that is to be run on a particular computer, and different classes of jobs are controlled at the start of the job (this will be explained in detail later). MCP uses the Client/Server to contact the database and request the next file, if any, that requires processing. All steps are retrieved at that time, and MCP

runs DSP, performs the processing. The command files used report status to the database, again using the Client/Server, and the next file is again requested. If there are no files ready for processing, MCP waits a specified time, then requests again.

2. Workstation DSP Image Processing System

DSP has been extended to run under UNIX, and many changes and enhancements have been made, including (but not limited to) many changes to the algorithms that calculate SST from AVHRR data, and remapping and DSP Image production.

3. DSP Command Procedure files

The DSP command procedure files used in the automatic processing have all been revised to be compatible with both the Workstation DSP and the new techniques associated with the automatic processing system.

4. Client/Server

This is a new capability. It was required for two reasons. To facilitate processing under UNIX, a method was needed to pass information between the VAX/VMS database interface and (theoretically) any operating system. Currently, the Client/Server operates between VMS and UNIX. In addition, a way to insure sequential access to the database was deemed valuable. It was noted during use of the old system that the combination of multiple contacts by one job, and multiple jobs, frequently caused problems, resulting in slow database access, and, at times, conflict between competing requests. Sequential access through the Client/Server, in addition to reducing the number of database contacts to two per job, should result in much greater efficiency during those contacts.

5. AUTOPROC Database

The database schema (construction and relationships between the database tables) has undergone extensive revisions and extensions in the past year. While the two major tables, MAIN and PROCESS_CONTROL retain their original purpose, the basic structure has been changed drastically, both to improve efficiency and to provide additional functionality not present in the original database.

Early in the year, a major database reorganization eliminated many of the tables resident in the old database. Previously, information in the MAIN and PROCESS_CONTROL tables was encoded using a series of look-up tables. For example, the SENSOR field in the MAIN relation contained only code numbers, and a SENSOR table held the actual sensor names. These look-up tables were, in general, replaced by direct storage of the sensor name in the MAIN table itself. While this expands the size of the database (as a 20-character SENSOR field must be stored for each MAIN record,) the access efficiency is greatly enhanced, because both table cross-queries are eliminated, and extraneous queries (to retrieve the coded keywords) are eliminated. As database access proved to be a major bottleneck during the CZCS processing, the increased database size seemed a reasonable trade-off to achieve increased database access efficiency.

While most old database fields have been retained (in some form) many more have been added, to accommodate new functionality.

6. .VMS AUTOPROC Database Interface

Virtually every element in the interface has been changed to some degree in the last year. Some changes, such as implementation of the Client/Server model, have been major, and some, such as changing the name of the GROUP table to USER_GROUP, have been minor. Many of the base subroutines used in the old system have been retained, albeit with changes. Additional subroutines and new 'contact point(s)' between the interface and the controlling/monitoring programs have been implemented.

Previously, there were only two types of 'contact points.' A special data ingest program was used to enter new records into the database. One program (DBCONNECT) and subroutine (DB_CONNECT) were used both to retrieve information from and update the database as needed. New record addition is still performed in a special ingest program, but the method of adding those records has changed almost entirely.

Previously, only one record was added to the MAIN and PROCESS_CONTROL tables in the database for each file to be processed. Now, multiple PROCESS_CONTROL records can be created for each MAIN record, and a method has been devised to automatically trigger one (or many) job(s) upon completion of one

(or many) other job(s). This method of triggering will be explained in greater detail in a later section.

This single contact point (DBCONNECT) previously used for both job requests and reports, has been divided into two separate functions in two programs/subroutines, DBREQUEST.MICE/DB_REPORT.F and DBREPORT.MICE/DB_REPORT.F. (A MICE file is a RATFOR, DSP program permitting easy, direct input of data into a program.) One major change in the request process is that rather than retrieving the processing steps one at a time, all process steps for a given job are retrieved at together, reducing the number of database accesses. In addition, many new subroutines have been added to the interface enhancing and extending its functionality.

A major change is the use of the Client/Server to provide a communication path between calling programs and the interface.

Previously, one program, MCP, which rang in batch mode, monitored the number of running jobs, contacted the database when new jobs were to be started, and spawned off batch jobs (called DBBAT) to perform the processing. Each DBBAT would independently contact the database, retrieves processing steps, and performs the processing. The database would be updated after each processing step was completed, and each DBBAT remained connected to the database during its lifetime. This resulted in slow database response and frequent lock conflicts, as multiple jobs attempted queries and update multiple times.

The large-scale structural reorganization has eliminated problems caused by multiple attachments/multiple updates. The use of the Client/Server mandates sequential access to the database, eliminating the possibility of lock conflict. Each job only involves two database contacts, first to retrieve the job steps, then to update the database after processing. The old subroutine that had been used for both purposes, DBCONNECT.F, has been replaces with two single purposes subroutines, db_Request.f and db_Report.f. These subroutines can be called either by the new UNIX C version of MCP (which uses the Client/Server), or by individual MICE programs which (currently) bypass the server (for testing purposes).

The changes in the addition of records to the database have been extensive. Previously, one MAIN record and one PROCESS_CONTROL record were used for each input file. Now, a suite of PROCESS_CONTROL records is added for each MAIN record, and each represents a piece of a processing thread. Individual records can trigger processing of other records, and classes of records can trigger other classes of records.

A number of tools have been developed which permit users to monitor and direct the database and processing activities, and more re planned. Currently, most of them do not work through the Client/Server, so users must be careful not to interfere with the automatic operations. These will soon be brought into the Client/Server realm.

7. Miscellaneous

A number of programs, VMS command files, SQL command files, DSP programs, DSP command procedure files, data transfer files, UNIX shell scripts have been added to facilitate data entry and retrieval, provide monitoring, and stop and start various pieces of the system.

B.1.2 December Activity Summary

Previously, two recipes (GAC_PTN & GAC_PTD, the time-bin procedures) had been treated as special cases, so only two jobs would run at one time. The new field PROCESS_CLASS, was added to the PROCESS_CONTROL table, and selection of job-limited records are based on this filed. The current fields are UNLIMITED, DAY, NITE and SPACEBIN, and the number of jobs of each class is controlled on the processing computer. The single recipe that had performed the ingest, atmospheric correction and spacebin, GINGEST, was broken into two separate recipes, GINGEST (ingest and atmos. corr.) and GSPACEBIN (space bin), so the number of spacebin jobs could be limited.

Another new field TRIGGER_REC was added to the PROCESS_CONTROL relation, to accommodate the GINGEST/GSPACEBIN recipes. At record addition, the GSPACEBIN record is added first, and the TRIGGER_REC is assigned a value of zero, and the PROCESS_STATUS is set to 'HOLD'. When the GINGEST record is added, its TRIGGER_REC is set equal to the PROCESS_CONTROL record of the GSPACEBIN job. After the

GINGEST job has finished, the PCR pointed to by the TRIGGER_REC will be marked as 'SUBMITTED' for processing.

A program/subroutine pair called COMPLETEMAIN.MICE and COMPLETE_MAIN.F were written to move finished records out of the PROCESS_CONTROL table into a new table called PC_COMLETE. Since most of the queries in the interface are to the PROCESS_CONTROL table, reducing its' size will make the process more efficient. A second program/subroutine pair, RESTOREMAIN.MICE and RECTORE_MAIN.F, reverses this, and moves PCRs back into PROCESS_CONTROL from PC_COMPLETE.

The new system was implemented and testing started.

B.2 Client/Server Status (S) (PATHFINDER)

The following items describe the creation and modification of the client/server implementation at RSMAS supporting the DSP system during 1992:

B.2.1 January through November Activity Summary

B.2.1.1. January - the early version of mcp has been implemented on VMS, however, it is in preliminary not operational form.

B.2.1.2. February -

a. Efforts begin to move mcp to UNIX environment.

b. The function that handles database, dbcontrol are separated from DSP.

B.2.1.3. March -

a. The initial client/server implementation becomes operational.
b. Queues similar to those available on VAXes are added to the UNIX mcp.
c. processmosaic was added to the processing steps as part of the test; at this juncture, one file can running manually.
d. The client/server is interacting with the main database.

B.2.1.4. April -

a. Due to information gathered in preliminary testing, changes to the client/server model were made:

1. Client implementation was not executing in the DSP environment, it was separated from DSP.

2. System philosophy was modified; mcp is given the task of contacting database. This leaves dbbat as a subordinate process with a single task. I.e. invoke DSP and run the command procedure specified by mcp.

b. Processing is being moved to the SGI 4D/480.

c. Testing of mcp and dbbat on UNIX continued; at this point mcp executes and fetches the job steps through client/server and put them in the specified directory. Both mcp and dbbat are running independently. Each suspends if no tasks are available and then checks to determine if there is a job to do.

B.2.1.5. May - A version of mcp has run multiple record tests.

B.2.1.6. June

a. Ingest inserted as a phony step in the database; the test executed as expected.

b. Two tests, CZCS and AVHRR GAC, are executing from the database and client/server perspective.

c. Server moved to a workstation.

d. System tests were executed successfully through the mosaic stage.

B.2.1.7. July

a. A client was created on VMS to allow ingest to be added to database as a step.

b. The new client added records to main and process_control tables.

c. Complete system tests executed successfully; every record in the database is marked correctly. Three days worth data were processed.

B.2.1.8. August

a. Testing continued. Problem areas that were discovered during system test were investigated.

b. After further testing, mcp was modified to execute processes in parallel.

c. Hurricane...

B.2.1.9. September

a. Testing continued using the parallel mcp implementation to determine problem sources, and to tune and optimize the system for efficiency and speed.

1. Test run of ten day's worth of data in a sequential manner.

2. System modification to facilitate concurrent processing.

3. Additional testing of concurrency improvements after modification.

b. After examining system performance and determining the causes listed above, we modified mcp and the database supporting programs to ensure greater concurrency.

c. After modification, we ran additional tests and obtained much better results. We have achieved an increase in concurrency and obtained some insight into avenues for further tuning the system.

d. The additional testing detected the source of the processing hangs that we had experienced; it was DSP/CALLER.

B.2.1.10. October

a. Continuous testing of the mcp structure continued by processing GAC data. The test consisted of executing eight mcp's [6 level 2 and 2 level 3] concurrently.

b. The tests exposed

1. processing bottlenecks in the system structure

2. shortcomings in DSP which needed to be addressed within a production environment.

c. DSP was modified as a result of the lessons learned and testing continued.

B.2.1.11. November

DSP/CALLER bugs were corrected, mcp executing properly.

B.2.2.1. December -

A. General SeaWIFS Activities

a. Generated equal area grid following ISCCP grid description

b. 9 KM fields, converted programs to use 4096 pixel scans

c. Built SPACEBIN and TIMEBIN programs

d. Built ANLY7 preliminary SeaWIFS LEVEL2 program, used to test processing timing and system flow

e. VDC description

integrated GORDON/WANG radiative transfer program to compute RAYLEIGH-AEROSOL interaction
initiated activity to compute ~10000 rte simulations

B. Direct SeaWIFS Support

a. Examining program supplied by C. Molyneaux of SGI, called vdc, to adapt it to operate in the DSP environment. vdc is performing rte processing on Modis

b. System support for vdc expanded by nfs mounting the necessary directories.

c. vdc adapted for DEC stations and VAX stations using different .vdc files for each machine.

d. Consulted with C. Molyneaux about current processing scheme; Chuck's idea was to use the same .vdc file for all machines.

e. vdc now executing in parallel on five DEC stations. At year's end, about 300 rte command files had been processed.

B.3 Matchup Database (P)

The following paragraphs list the 1992 activities directed towards the development of a matchup database of AVHRR and in situ observations. The matchup database is to be used in the development and validation of new algorithms to estimate sea surface temperature (SST) from AVHRR brightness temperatures. As each of the accomplishments was described in detail in the monthly reports, they are summarized here. The main accomplishments were:

• Compilation and reformatting of in situ environmental observations from moored buoys and drifting buoys. This was done in cooperation with colleagues at NASA’s Goddard Space Flight Center.

• Implementation and testing of a flexible methodology to determine which in situ observations were viewed by the AVHRR within a given time/space window. This methodology (the “time of closest approach," or TCAP) allowed us to streamline the process of extracting AVHRR/GAC data corresponding to the in situ times/locations. The methodology is flexible and, provided orbital and scanner models are available, it could be applied to other sensors such as SeaWIFS, which will also require matchups for algorithm validation.

• Implementation and testing of software to extract appropriate data from the AVHRR/GAC optical disks at the times and locations of in situ data. The procedures were tested by producing a one-month global matchup database for January 1988.

• Construction of "experimental" matchup databases based on AVHRR data archived at Miami and a limited number of moored buoys off the US east coast and Gulf of Mexico. The multi-year (1987-1989), multi-sensor (NOAA-9 and NOAA-11) database had two applications:

- It gave us "hands-on" experience in the development of the procedures involved in the construction of the global, operational matchup database. Several modifications to our approaches resulted from experience gained using the experimental database.

- The experimental database was used to develop and test initial SST algorithms and procedures for data-quality flagging.

• Incorporation of data from satellite sensors other than the AVHRR to the matchup database. We experimented with the use of integrated atmospheric water vapor concentrations derived from the SSM/I, a microwave radiometer. Procedures are currently being tested to extract water vapor concentrations at matchup times/locations.

In summary, this year gave us direct experience in building a matchup database of satellite and in situ data. The construction of a global matchup database for 1988 is currently ongoing, and will be completed towards the end of February 1993.

B.4 DSP Support (H)

B.4.1 The following section provides summary of DSP support activities from January through November 1992:

DSP was modified so that it would run on the following machines: VAXstation, DECstation, Sun, ALPHA (UNIX), SGI, Intel 386, MacII, RS/6000, Next, and Stardent.

Some of the major features we added to DSP:

Unique image plane header files for each user.
16 planes per user.
Shared regions for WRKSPC and GETCOM.
Increased maximum pixels per line to 4096.
Improved error handling and reporting.

Some fixes to DSP:

Fixed the problem that caused users to see other users workspace variables.
Fixed the problem of CALLER crashing when there were too many messages.
The workstations can read old-style image files.

Some new programs:

TCAP - input date, time, and location; output satellite time, orbit, and distance from location.
PATHSST, PATHBIN, PATHTIME, PATHMOS, PATHMAP - New atmospheric correction, 9 km binning and a mosaic that remaps.
Programs to control automatic processing on multiple machines using an RDB database.

B.4.2 The following section provides summary of DSP support activities in December 1992.

B.4.2.1 Testing:

Using client/server/rdb database to process GAC data on SGI.
Testing new versions of DSP - separate header files for each users, shared region for workspace, 16 planes.

B.4.2.2 Modifications/Additions to DSP:

Misc. source convention changes for ALPHA (new IMGFILE type).
Misc. source convention changes for SUN. New program PATHMAP remaps PST files into image files.
New library routine: Dsp_DeleteFile
Consolidate GETCOM's global variables into one common area.
Add land bitmask to PATHBIN.
Misc. changes to make CALLER portable.
Add shared memory region on UNIX (GETCOM).
Add threshold option to only do COMPOS if pixels are different by threshold amount.

B.4.2.3 Problems fixed:

Change source to work with new FORTRAN compilers (decode formats and type casting).
Misc. changes for the VMS C compiler.
Fix SHOCOM for case when there is no frame buffer.
Fix CDR for SGIs.
Fix SCRIPP to handle Ocean Imaging tapes properly.
Misc. changes to make TROUTC portable.
Fix output device check and row and column bounding check in PATHMOS.

B.5 PATHFINDER Activities

B.5.1 Robert Evans attended the AGU meeting in San Francisco. He presented a paper by Robert Evans and Guillermo Podesta titled "The NOAA/NASA Ocean Pathfinder Project - Generation of a consistent 10 year global SST data set using the NOAA AVHRR sensors." Bob discussed the Project, the Objectives, the Science Working Group (functions and membership), the need for match-up databases, the process of matching-up satellite and in situ observations, the AVHRR Pathfinder Ocean Products, Community Participation, NOAA-9's AVHRR from '85-'89, the global match-ups and the regional SST sectors.

As part of Community Participation, he talked about the JPL DAAC's production role, about an "Exploratory" data set of global 9 km equal-area fields (2-3 months worth from '88) which will be available for evaluation in February 1993, a global match-up database available in February 1993, and asked for ideas from the community about SST algorithms and data quality assessment. Further, volunteers were requested to examine test months;

B.5.2 Additional Discussions

B.5.2.1 Robert Evans, Andy Tran and Liz Smith discussed the "Exploratory" Pathfinder Ocean data set to be released in February, and about Miami's status since our meeting in November. For the "Exploratory" data set, we (tentatively) plan to provide 2-3 months of data from '88 that will be approximately an exact replicate of the current MCSST data set. This is because most of the "guinea pigs" (Peter Cornillon, Richard Legekis, Dick

Reynolds, Ted Strub, John Bates, and Dave Halpern) are familiar with that format. Read software and documentation will be provided.

Discussions were held concerning a meeting of ocean science working group to review progress and discuss what remains prior to starting 1988 part of benchmark period.

The merits of the cloud detection methods have been discussed.

B.5.3 Other Activities

B.5.3.1 A mosaic program based on remap was created to project the equal-area 9 km binned data files into standard earth projections. The program, mapsst, was used to prepare daily day and night 9 km Robinson projection images for the AGU meeting.

B.5.3.2 Twenty-one days of GAC-IR data for 1988 has been processed. Currently a 30 day test is beginning and the results are being transferred to URI for use and testing with their cloud rejection programs and to JPL for conversion to pathfinder distribution formats.

B.6 SeaWIFS Support

B.6.1 Jim Brown has been working with Gene Feldman's group and SGI to get an optical disk capability on the SGI machines.

B.6.2 Dalu worked with SGI's Chuck Molyneaux to get vdc working in a multiple machine, multi-vendor environment to run Gordon's radiative transfer codes to generate the SeaWIFS Rayleigh-aerosol interaction coefficients. Approximately 10,000 1-2 hour runs are planned for the initial set of aerosol phase functions. We have received the initial set of 2000 jobs from Gordon/Wang. Hardware (disk and network) has been installed to facilitate the calculations.

B.6.3 A 9 km CZCS/SeaWIFS version of the mapping program is being generated using mapsst as a basis.

B.6.4 The core (workspace, GETCOM) DSP system has been converted to use mapped regions rather than messaging to exchange control

and environment information. This change was introduced to increase reliability and execution speed.

B.7 MODIS Support

B.7.1 The upgrade to SGI 480 server machine (R3000 CPU) to next generation processor, R4K (R4000 processor) has been ordered. These processors will be utilized for radiative transfer calculations and for working with pathfinder and SeaWIFS fields.

B.7.2 All machines have been converted to use both FDDI and Ethernet communication channels. Some problems were experienced with SGI machines and we have requested and installed new hardware/software.

B.7.3 FDDI networking: received FDDI interfaces

B.7.4 Disk Subsystems: added disk capacity, product fields for SeaWiFS and AVHRR that require ~ 0.5 gigabytes.

B.7.5 MODIS:

B.7.5.1 Established initial computer facility to support the MODIS ocean team computer facility (MOTCF). MOTCF is the basis of implementation and processing for Pathfinder and SeaWiFS.

B.7.5.2 Robert Evans attended MODIS ocean team meetings in Miami, at GSFC, and Santa Barbara

B.7.5.3 Discussions were held with ocean team members concerning integration of level2 and product algorithms. due to the need to incorporate fields outside of the MODIS data stream, we are examining the possibility of implementing selected product algorithms using level 3 MODIS products.

B.8 Team Interactions

C. FUTURE ACTIVITIES

C.1 Database Future Work

C.2 Client/Server Future Work

C.2.1. Creation of a resource manager and a performance monitor.
C.2.2. Expansion of the error handler to provide broader coverage and to integrate into the overall system error recovery scheme.
C.2.3. Continue testing the client/server with CZCS and AVHRR data. This would include the acquisition of a UNIX resident database to run parallel tests.
C.2.4 Continue enhancement of processing efficiency through greater use of concurrent processing.
C.2.5 Examine other processing schemes to determine which elements could be either included or adapted for use within the client/server concept.

C.3 Pathfinder (P)

C.3.1. Continue development of linking processes between *in-situ* and processed satellite data.
C.3.2. Expand the validation dataset.

C.4 Headquarters (H)

C.4.1. Create tools to assist in results interpolation.
C.4.2. DSP - Fix programs that access the graphics plane to use the navigation from the input image and not the graphics plane.
C.4.3 Refine PATH binning and mosaic pixel quality algorithm to eliminate clouds.
C.4.4 Verify workstation DSP (SGI, SUN, DECstation, VAXstation) by comparing each program's output with the Adage system.

C.5 Modis (M)

C.5.1. Continue working with H. Gordon on an implementation of prototype ocean color atmospheric correction algorithms.
C.5.2 Continue working with D. Clark on *in-situ* database requirements.

D. PROBLEMS

D.1 Database Problems

None listed.

D.2 Client/Server Problems

None listed.

D.3 Matchup Database Problems

None listed.

D.4 DSP And Headquarters Related Problems

None listed.

Appendix 1:Equal-area Gridding Scheme Proposed for SeaWIFS Ocean Color Products

App.1.1 Introduction

This document describes the equal-area gridding scheme proposed by the RSMAS Remote Sensing Group for the SeaWIFS binned ocean color products. The same approach is being adopted for sea surface temperature fields produced by the AVHRR Pathfinder project. The gridding scheme is based on that adopted by the International Satellite Cloud Climatology Project (ISSCP).

This document does not motivate the need for an equal area grid for SeaWIFS or other oceanographic products. Such motivation can be found in a paper by W. Rossow and L. Gardner (Selection of a map grid for data analysis and archival, *Journal of Climate and Applied Meteorology*, 1984, 23: 1253-1257). Furthermore, this document describes only the design of the proposed equal-area grid, and does not discuss other related topics such as rules for spatially or temporally combining observations into the equal-area bins.

App.1.2 Overview

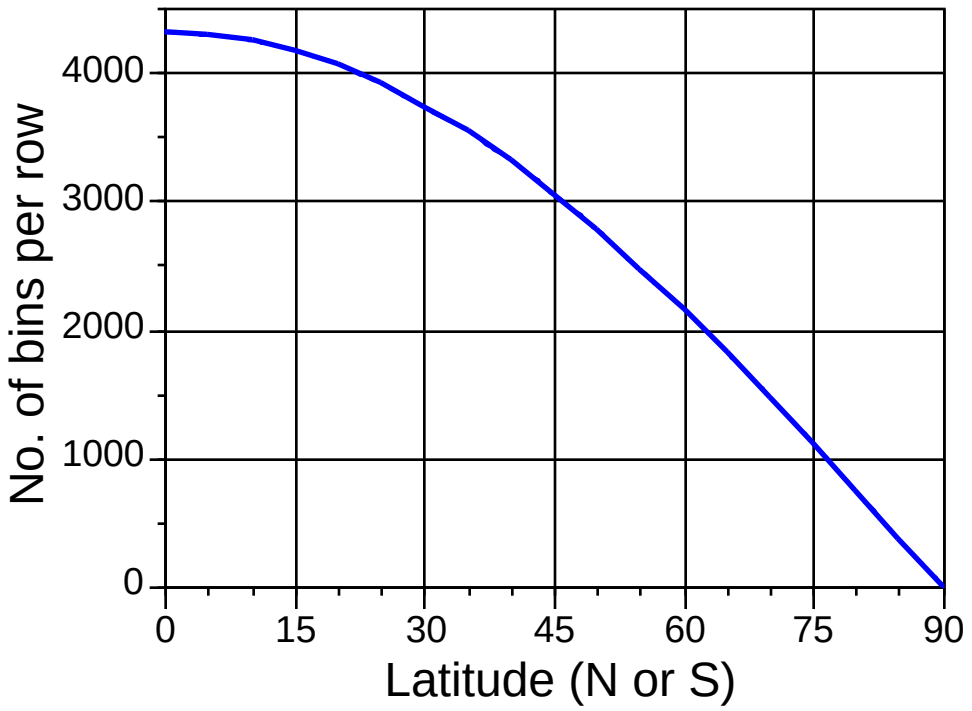
The gridding scheme proposed consists of rectangular bins or tiles, arranged in zonal rows. A compromise between data processing and storage capabilities, on one side, and the potential geophysical applications of satellite data, on the other side, suggest that a suitable minimum bin size would be approximately 8-10 km on a side.

In the scheme proposed here, the tiles are approximately 9.28 km on a side. This size (9.28 km) was chosen because (a) it has approximately the desired minimum resolution, and (b) it results in 2160 zonal rows of tiles from pole to pole (i.e., 1080 in each hemisphere). This particular number of rows (2160) has some

advantages which will be discussed in more detail below. Because the total number of rows is even, the bins will never straddle the Equator (i.e., there will be an equal number of rows above and below the Equator). This avoids possible situations where the Coriolis factor is zero. This is a characteristic that numerical modellers expect from any gridding scheme adopted.

The total number of approximately 9-km bins is 5,940,422. The bins or tiles are arranged in a series of zonal rows; the number of tiles per row varies. The rows immediately above and below the Equator have 4320 tiles. This number is derived by dividing the perimeter of the Earth at the Equator by the standard tile size (i.e., $2\pi R_e/9.28$), where R_e is the equatorial radius of the Earth (R_e = 6378.145 km). The number of tiles per row decreases approximately as a cosine function as the rows get closer to each pole (rigorously, there should be an adjustment for ellipticity of the Earth, as the equatorial radius decreases progressively to the smaller polar radius; this adjustment is not applied in the current implementation). At the poles, the number of tiles is always three. This special situation will be discussed in detail below. The number of tiles per row as a function of latitude is shown on Figure 1.

Figure App1.1. Number of 9.28 km tiles per zonal row as a function of latitude (North or South). The number of tiles is 4320 at the Equator and decreases to 3 at the poles.



The number of bins in each zonal row is always an integer. To ensure an integer number of bins, the width of each bin (the size of a bin along a parallel, or x-length) must vary slightly from row to row. However, the bins are always 9.28 km long along the meridians. That is, only one of the bin dimensions changes. The size of the bins at each zonal row is established in the following manner. First, a preliminary value for the number of tiles (N_p) at a given latitude (L) is computed as

$$N_p = 2\pi r / X,$$

where X is the x-size of a bin at the Equator (9.28 km) and r is the radius of the circle produced by slicing the Earth with a plane parallel to the Equator at latitude L. The radius r can be calculated as

$$r = R_{eq} \cos(L),$$

where R_{eq} is the equatorial radius of the Earth. If the fractional part of N_p is greater or equal than 0.5, then N_p is rounded up to the nearest integer (i.e., the final number of tiles will be the integer portion of N_p plus one), otherwise N_p is rounded down (the final number of tiles is the integer portion of N_p. Once the final integer number of tiles along a row is calculated, the X-size of the tiles must

be adjusted. This is done by dividing the perimeter of the row ($2\pi r$) by the integer number of tiles. The result is the x-length of a tile (width) for a given row.

Because the x-length of the tiles is adjusted to ensure an integer number at each row, the “equal area” characteristics of this binning scheme are not rigorously preserved. However, variations in tile size are negligible throughout most of the globe, and only become relevant at very high latitudes, where there are fewer tiles per row and, thus, any adjustments are more noticeable. As soon as the number of tiles increases with distance from the poles, the difference between tile sizes rapidly becomes practically unnoticeable. To provide an idea of the magnitude of the fluctuations in tile size, the worst possible case occurs when half a tile remains “uncovered” after filling a zonal row with an integer number of tiles. Once a row has 100 bins (approximately 16 rows, or 148 km from the poles), the worst possible difference between the actual tile x-length and the standard x-length is of the order of 0.5% (i.e., half a tile's length redistributed among about 100 tiles). For a tile of about 9 km a side, this represents a difference in the x-length of about 45 m. Through a similar calculation, a row with 50 bins (about 80 km away from the poles) has a 1% variation with respect to the standard bin size.

The gridding scheme described here has an extremely useful feature: the number of 9.28 km tiles in each hemisphere (1080) is divisible by many numbers (e.g., 2,3,4,5,6) and therefore it is extremely easy to generate an integer number of rows at many useful spatial resolutions. For instance, 12 rows of 9.28 km tiles can be combined to generate zonal bands of approximately one degree (one degree of latitude is equal to 111.12 km; 12 bins would form a band 111.20 km wide). Another example is the use of 30 rows of to generate zonal bands of approximately 2.5° (a typical output resolution of atmospheric circulation models).

App.1.3 The poles

Both the North and South poles are special cases in the gridding scheme presented here. The pole areas are always covered by three tiles, shaped like pie sectors. While the meridional size of the polar bins (the y-length) will be the usual 9.28 km, the length of the bins along the arc of the sectors will be slightly larger. Neglecting sphericity, the area encompassed by the last row of tiles is πX^2, where X = 9.28 km. If we express the area of the circle as a rectangle of height X, the remaining dimension is πX. If we divide the perimeter by three (to yield three tiles), each tile will have dimensions X by $\pi X/3$ (approximately 1.05X). That is, the bases of the triangular polar tiles are about 5% larger than the x-length of the equatorial tiles.

App.1.4 Binning software

Two main routines have been developed to perform the two principal transformations required for binning and mapping data. The first routine converts latitudes and longitudes into bin numbers **(fwdld)**. The second routine performs the inverse transformation, that is, given a bin number it returns a latitude and longitude corresponding to the centroid of that bin **(revld)**. These two routines use a common initialization routine that must be executed prior to calling the conversion routines **(bin9kminit).** Annotated code for these routines is provided below.

```
# Copyright 1988-1992 by
#Rosenstiel School of Marine and Atmospheric Science,
# University of Miami, Miami, Florida.
#
#                   All Rights Reserved
#
# Permission to use, copy, modify, and distribute this software and
its #documentation for non-commercial purposes and without fee is
hereby granted, #copyright notice and this permission notice appear
in supporting documentation, #and that the names of University of
Miami and/or RSMAS not be used in #advertising or publicity
pertaining to distribution of the softwarewithout specific, #written
prior permission.
#
# UNIVERSITY OF MIAMI DISCLAIMS ALL WARRANTIES WITH
REGARD TO #THIS SOFTWARE, INCLUDING ALL IMPLIED
WARRANTIES OF #MERCHANTABILITY AND FITNESS, IN NO EVENT
SHALL UNIVERSITY OF #MIAMI BE LIABLE FOR ANY SPECIAL,
INDIRECT OR CONSEQUENTIAL #DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS #OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, #NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN #CONNECTION WITH THE
USE OR PERFORMANCE OF THIS SOFTWARE.
##
#
#  bin9kminit.rat
#
#  This subroutine initializes structures that are needed by the
#  routines that convert between bin number and location (latitude,
#  longitude).  This routine must be called once before the
conversion
#  routines are used.
#
#  The argument to this subroutine is the total number of rows of
tiles
#  from pole to pole.  This number must be less than or equal to
4320.
#  For "9km" bins this number should be 2160.
#
#  Bin numbers start at one.  The first bin starts at the Greenwich
#  Meridian and goes east to 120 Deg.  There are three bins in the
first
```

```
#   and last rows (at each pole). The bin numbers increment to the
east
#   around the globe and then north to the next row.
#   The last bin is from 240 Deg to 360 Deg at the North Pole.
#

include "miami"                      # Ratfor definitions

define      Re    6378.145d0            # Earth radius (km)
[Equator]
define      Pi    3.1415926535897 93d0      # Circular constant

define(Rad(x),((x)*Pi/180d0))             # degrees to radians
define(Deg(x),((x)*180d0/Pi))             # radians to degrees

ifdef(unix)
define(cosd(x),dcos(Rad(x)))        # cos(Angle in degrees)
enddef

subroutine bin9kminit(Rows)

int    Rows             # Total number of rows in the grid

int    k                # local temporary variable
double      blat              # base latitude for the row
double  clat                 # center latitude for the row

include "bin9km_com"        # declare shared arrays

init   INITED / $F /           # Check to ensure this routine is called
once

      #
      # Make sure the input parameter, Rows, is a valid number.
      #

      if (Rows > MAX_ROW)
         stop 'bin9kminit: Rows too large'

      #
      # Calculate constants for this grid (defined by Rows).
      #
```

```
      # Calculate the height (and width) of each row
      # GRDSIZ for "9km" bins is approximately 9.28
      GRDSIZ = Pi*Re/Rows;              # Bin spacing for this grid

      # MAXROW is the actual number of rows in this grid
      MAXROW = Rows                     # total rows in mapping

      #
      # Initialize arrays that are used by the conversion routines
      #

      # The first bin for the first row is numbered one
      BASBIN(1) = 1

      # For each row in the grid
      do k=1,MAXROW {

         # calculate the base (southernmost) latitude for the row
         # base lat = (number of previous rows) * (the size of a row)
         # The size of a row: Deg(GRDSIZ/Re) is the same as
(180/Rows)
         # Subtract 90 Deg so that the lat's range from -90 to 90.

         blat = (k-1)*Deg(GRDSIZ/Re)-90d0     # compute base
latitude

         # calculate the latitude at the center of the row
         # center lat = base lat plus half the size of a row
         clat = blat + Deg(GRDSIZ/Re)/2d0      # center latitude of
row

         # calculate the number of bins in the row
         # number of bins = (earth circumference at center lat)/(bin
size)
         NUMBIN(k) = 2*Pi*(Re/GRDSIZ)*cosd(clat) + 0.5

         # compute the bin number of the first bin in this row (the
base bin)
         # first bin = previous rows first bin plus number of bins
         if (k > 1)               # already set first base bin to one
            BASBIN(k) = BASBIN(k-1) + NUMBIN(k-1)

         LATBIN(k) = sngl(clat)     # center latitude for this row
```

```
	}

	# The BASBIN array is one element larger than the number of
rows to
	# simplify a check in the rev1d routine.

	BASBIN(MAXROW+1) = BASBIN(MAXROW) +
NUMBIN(MAXROW)
	N0 = BASBIN(MAXROW+1)-1		# Total bins in the
whole grid

	# Set this flag (INITED) so that fwd1d and rev1d can check to
make
	# sure the arrays have been initialized.

	INITED = $T			# Initialization complete

end

subroutine fwd1d(alon,alat,N)

#  Convert earth location (latitude, longitude) to grid coordinate (bin
number)

single	alon   # input longitude, may be 0..360 or -180..180
single  alat # input latitude, -90..90
long   N      # output bin number, 1..N0

int    k      # local variable used for the row number
double	blon   # local variable used to change lon to 0..360

include "bin9km_com"		# declare shared arrays

	# Only continue if the arrays have been initialized

	if (!INITED) stop 'fwd1d: bin9kminit not called'

	# Calculate the number of whole rows below the given latitude
	# row number = lat (in 0..180 range) times the number of rows
per Deg
	# [(GRDSIZ/Re) is the same as (Pi/Rows)]
	# add one and truncate to get the integer row number for this
lat
```

```
      k = 1+Rad(90d0+alat)/(GRDSIZ/Re)

      # Make sure the longitude is between 0 and 360

      blon = alon
      if (blon < 0d0)
         blon = blon + 360d0        # 0..360
      else if (blon >= 360d0)
         blon = blon - 360d0

      # bin number = (first bin in the row) plus
      #    (number of bins in the row) times (the fraction of the row
to the
      #     specified longitude)
      N = BASBIN(k) + NUMBIN(k)*(blon/360d0)

end

subroutine rev1d(N,alon,alat)

# Convert grid coordinate (bin number) to earth location (longitude,
latitude).
# This routine uses a binary search through BASBIN array (first bin
in each
# row) to find the row number for the bin.  Then the lon and lat are
# calculated.  The row number from the previous call is checked to
speed
# sequential requests.

long  N           # input bin number, 0..N0
single      alon        # output center longitude for the bin (-
180..180)
single  alat        # output center latitude for the row (-90..90)

int    nhi, nlo, nmid  # local variables used for the binary search
int    k           # local variable used for the row number
double      blon        # local variable for the computed longitude in
radians
double  clat            # local variable for the computed latitude

int    kold        # remember the row number between calls to this
routine
```

```
save  kold

include "bin9km_com"

init   kold / 0 /   # For the first time this routine is called, there is
                    # no previous row, it must be found by searching.

       # Make sure the arrays are initialized by bin9kminit

       if (!INITED) stop 'rev1d: bin9kminit not called'

       # Before doing the binary search for the row number, see if
this bin
       #   is in the same row as the previous bin was.

       if (kold > 0 &&
          BASBIN(kold) <= N && BASBIN(kold+1) > N) {
          k = kold                # use last value
       }
       else {
          # We have to search for the row

          # make sure the bin number is valid
          if (N < 1)
              N = 1               # south pole
          else if (N > N0)
              N = N0                      # north pole

          nlo = 1                 # binary search for k (row number)
          nhi = MAXROW                #  in range [1..MAXROW]
          repeat {
              nmid = (nhi+nlo+1)/2
              if (BASBIN(nmid) > N)
                 nhi = nmid-1
              else
                 nlo = nmid
              if (nlo == nhi) {
                 k = nlo
                 break
              }
          }
          kold = k          # cache for next time this routine is called
       }
```

```
      clat = LATBIN(k) # return the center lat for the row

      # compute the center longitude of the bin on the row
      # blon = (radians in a circle) times (the part of the circle from
the
      #   start (base) of the row to the middle of the specified bin)
      blon = 2*Pi*(N-BASBIN(k)+0.5)/NUMBIN(k)

      alat = clat          # return the center lat for the row
      alon = Deg(blon)  # change the lon to degrees

end
```

```
##
# Copyright 1988-1992 by
# Rosenstiel School of Marine and Atmospheric Science,
# University of Miami, Miami, Florida.
#
#                   All Rights Reserved
#
# Permission to use, copy, modify, and distribute this software and
its #documentation for non-commercial purposes and without fee is
hereby granted, #copyright notice and this permission notice appear
in supporting documentation, #and that the names of University of
Miami and/or RSMAS not be used in #advertising or publicity
pertaining to distribution of the softwarewithout specific, #written
prior permission.
#
# UNIVERSITY OF MIAMI DISCLAIMS ALL WARRANTIES WITH
REGARD TO #THIS SOFTWARE, INCLUDING ALL IMPLIED
WARRANTIES OF #MERCHANTABILITY AND FITNESS, IN NO EVENT
SHALL UNIVERSITY OF #MIAMI BE LIABLE FOR ANY SPECIAL,
INDIRECT OR CONSEQUENTIAL #DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS #OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, #NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN #CONNECTION WITH THE
USE OR PERFORMANCE OF THIS SOFTWARE.
##

# bin9km common
# This include file declares the arrays that are initialized by
bin9kminit
# and used by the rev1d and fwd1d conversion routines.

ifnotdef(MAX_ROW)
define      MAX_ROW 2160*2                  # rows from pole to
pole
enddef

flag   INITED                       # Initialize routine called
int    MAXROW                       # Total number of rows in grid
long   N0                       # Total number of bins in grid
double      GRDSIZ                          # Area of each bin (km)
share /BIN9KMFIX/ GRDSIZ, N0, MAXROW, INITED
save /BIN9KMFIX/
```

```
int    NUMBIN(MAX_ROW)                # Number of bins in each
row
long  BASBIN(MAX_ROW+1)        # First bin number in each row
single       LATBIN(MAX_ROW)                  # Center latitude for
each row
share /BIN9KMVAR/ NUMBIN, BASBIN, LATBIN
save /BIN9KMVAR/
```